

Algorithm: Lab1 (By Sujiv Shrestha ID:610145)

Problem 1. Which of the following functions are increasing? eventually nondecreasing? If you remember techniques from calculus, you can make use of those.

a. $f(x) = -x^2$

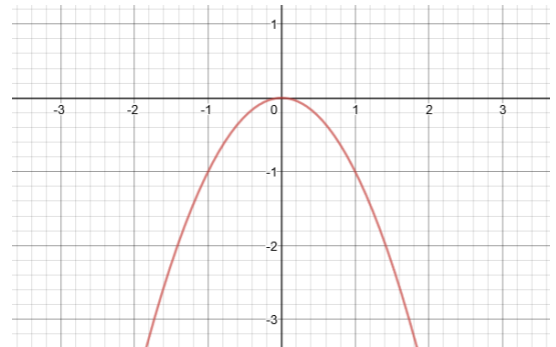
Answer: The function is increasing for the range $-\infty < x < 0$ and decreasing for the range $0 < x < \infty$

$$df/dx = -2x$$

Hence,

For $x < 0$, $df/dx > 0$, Increasing

For $x > 0$, $df/dx < 0$, decreasing



b. $f(x) = x^2 + 2x + 1$

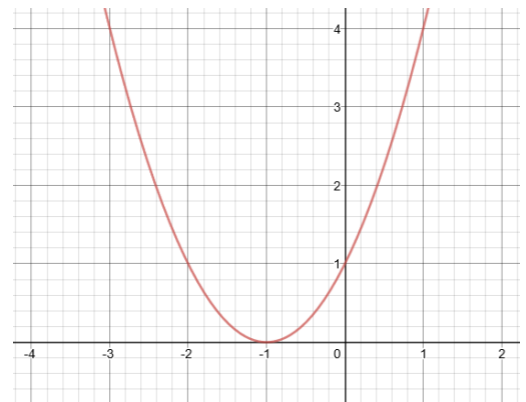
Answer: The function is increasing from the points $x = -1$ and onwards.

$$df/dx = 2x + 2$$

Hence,

For $x < -1$, $df/dx < 0$, decreasing

For $x > -1$, $df/dx > 0$, increasing

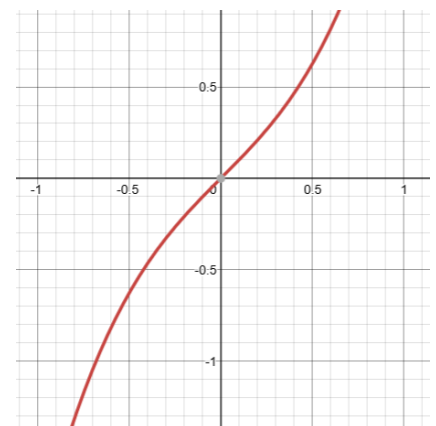


c. $f(x) = x^3 + x$

Answer: The function is increasing because the function gives higher value for all higher values of x .

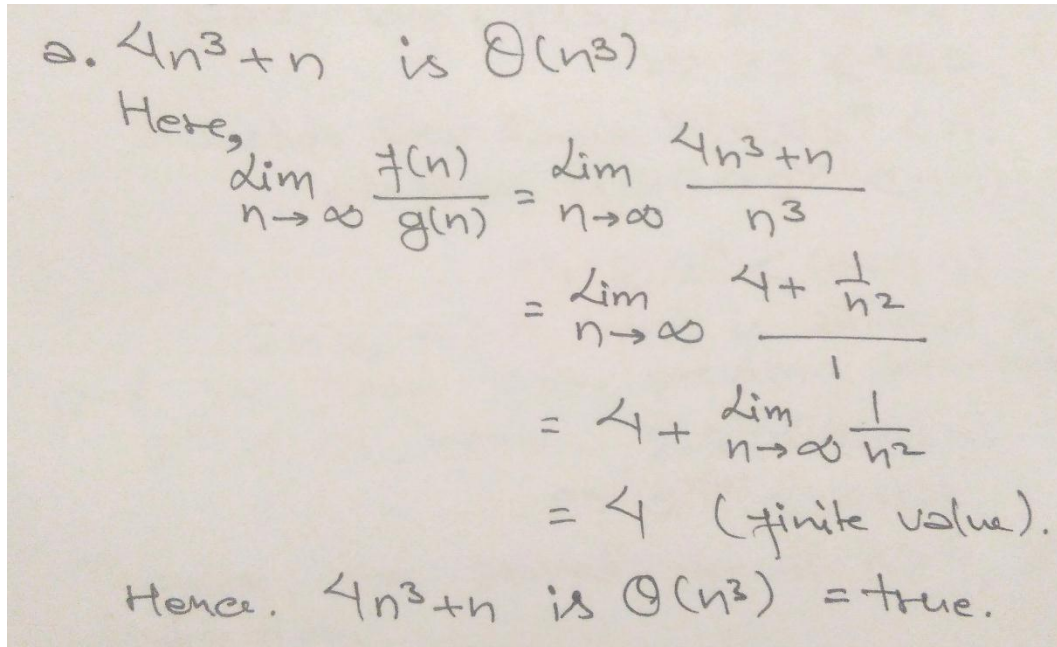
$$df/dx = 3x^2 + 1$$

Hence for any value of x , $df/dx > 0$ So the function is increasing



Problem 2. Use the limit definitions of complexity classes given in class to decide whether each of the following is true or false, and in each case, prove your answer.

a. $4n^3 + n$ is $\Theta(n^3)$



a. $4n^3 + n$ is $\Theta(n^3)$

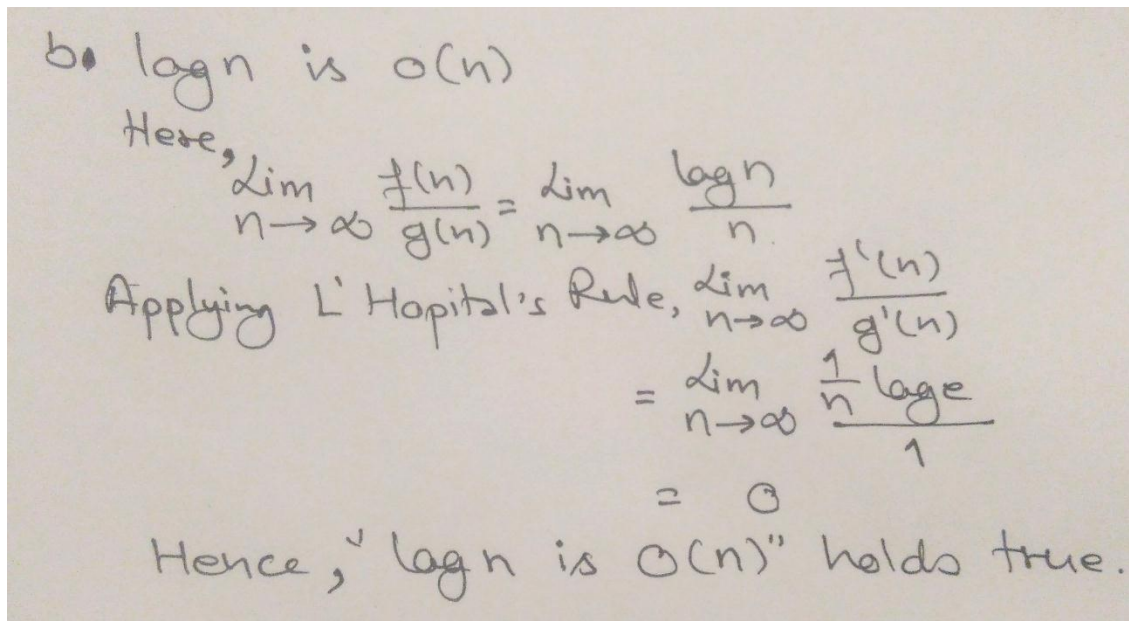
Here,

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{4n^3 + n}{n^3}$$
$$= \lim_{n \rightarrow \infty} \frac{4 + \frac{1}{n^2}}{1}$$
$$= 4 + \lim_{n \rightarrow \infty} \frac{1}{n^2}$$
$$= 4 \text{ (finite value).}$$

Hence, $4n^3 + n$ is $\Theta(n^3)$ = true.

Note: polynomials have same order of 3.

b. $\log n$ is $o(n)$



b. $\log n$ is $o(n)$

Here,

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{\log n}{n}$$

Applying L'Hopital's Rule,

$$\lim_{n \rightarrow \infty} \frac{f'(n)}{g'(n)}$$
$$= \lim_{n \rightarrow \infty} \frac{\frac{1}{n} \log e}{1}$$
$$= 0$$

Hence, " $\log n$ is $o(n)$ " holds true.

c. 2^n is $\omega(n^2)$

c. 2^n is $\omega(n^2)$.

Here, $\lim_{n \rightarrow \infty} \frac{g(n)}{f(n)} = \lim_{n \rightarrow \infty} \frac{n^2}{2^n}$

Applying L'Hopital's Rule, $\lim_{n \rightarrow \infty} \frac{g'(n)}{f'(n)}$

$$= \lim_{n \rightarrow \infty} \frac{2n}{2^n \ln 2}$$

Applying L'Hopital's Rule again,

$$= \lim_{n \rightarrow \infty} \frac{2}{2^n \cdot \ln 2 \cdot \ln 2}$$
$$= \lim_{n \rightarrow \infty} \frac{1}{2^n} \cdot \frac{2}{(\ln 2)^2}$$

Therefore, 2^n is $\omega(n^2)$ holds true i.e. 2^n grows much faster than n^2 .

d. 2^n is $o(3^n)$

d. 2^n is $o(3^n)$.

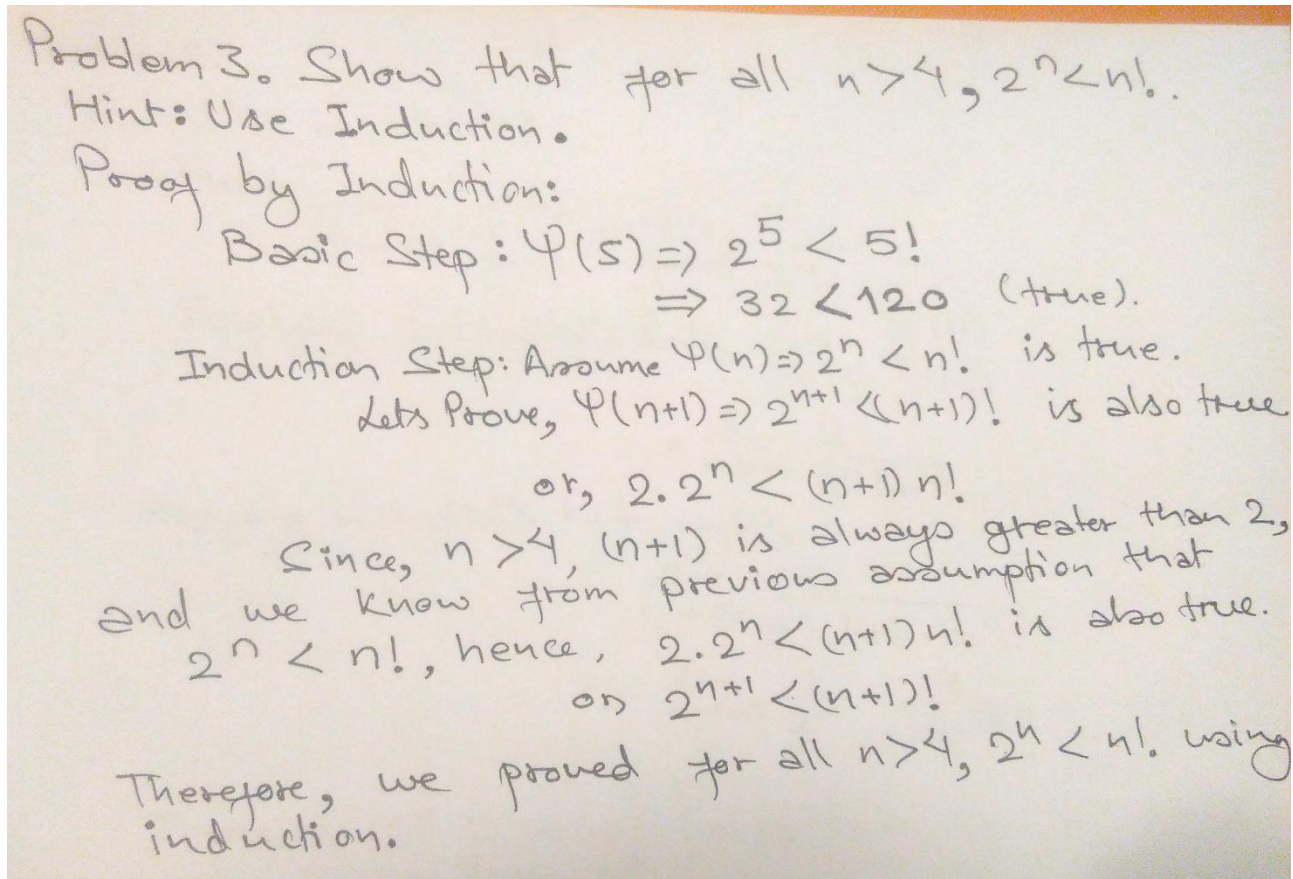
Here, $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \lim_{n \rightarrow \infty} \frac{2^n}{3^n}$

$$= \lim_{n \rightarrow \infty} \left(\frac{2}{3}\right)^n$$
$$= \lim_{n \rightarrow \infty} \frac{1}{\left(\frac{3}{2}\right)^n}$$

we know, $\frac{3}{2} > 1$. so $\lim_{n \rightarrow \infty} \frac{1}{\left(\frac{3}{2}\right)^n} = 0$.

Hence, 2^n is $o(3^n)$ holds true.

Problem 3. Show that for all $n > 4$, $2^n < n!$. Hint: Use induction.



Problem 4. GCD Problem: Given two positive integers m , n , is there a positive integer d that is a factor of both m and n and that is bigger than or equal to every integer d' that is also a factor of m and n ?

Write a Java method `int gcd(int m, int n)` which accepts positive integer inputs m ; n and outputs the greatest common divisor of m and n .

Examples

- If $m = 12$ and $n = 42$, return 6
- If $m = 7$ and $n = 9$, return 1

Solution=>

```
public static int GCD(int m, int n) {  
    return (n==0)?m:GCD(n, m%n);  
}
```

Problem 5. Implement the following Java method.

```
public static int secondSmallest(int[] arr) {  
    if(arr==null || arr.length < 2) {  
        throw new IllegalArgumentException("Input array too small");  
    }  
    //implement  
}
```

This method returns the second smallest element of the input array.

Examples

- If input is [1, 4, 2, 3], return 2.
- If input is [3, 3, 4, 7], return 3. (Smallest is 3, and second smallest is 3.)
- If input is [9], your program will throw an exception.

Solution=>

```
public static int secondSmallest(int[] arr) {  
    if(arr==null || arr.length < 2) {  
        throw new IllegalArgumentException("Input array too small");  
    }  
  
    int firstSmallest = arr[0];  
  
    int secondSmallest= arr[1];  
  
    for(int i=1;i<arr.length;i++) {  
        if(firstSmallest>arr[i]) {  
            secondSmallest = firstSmallest;  
            firstSmallest = arr[i];  
        }  
        if(secondSmallest>arr[i]&&arr[i]>firstSmallest) {  
            secondSmallest = arr[i];  
        }  
    }  
    return secondSmallest;  
}
```

Problem 6. SubsetSum Problem: given a set $S = \{s_0, s_1, s_2, \dots, s_{n-1}\}$ of positive integers and a non-negative integer k , is there a subset T of S so that the sum of the integers in T equals k ?

Formulate your own procedure for solving the SubsetSum. Think of it as a Java method `subsetsum` that accepts input S and k , and outputs a subset T of S with the property that the sum of the elements in T is k if such a T exists, or null if no such T can be found.

Examples

- If S is $[1, 3, 9, 4, 8, 5]$ and $k = 21$, return $[9, 4, 8]$ (since $9 + 4 + 8 = 21$)
- If S is $[1, 3, 9]$ and $k = 5$, return null (since no such subset can be found)
- If S is $[1, 3, 9, 4, 8, 5]$ and $k = 0$, return $[]$ (since the sum of the empty set is 0)

```
public static int[] subSetSum2(int[] arr, int sum) {
    if(sum==0)
        return new int[0];
    long size = (long) Math.pow(2, arr.length);
    int[] arr2=new int[arr.length];
    for(int i=0;i<size;i++) {
        int sum2=0;
        String array="";
        int count = 0;
        for(int j=0;j<arr.length;j++) {
            if((i&(1<<j))>0) {
                sum2+=arr[j];
                array=array+" "+arr[j];
                arr2[count] = arr[j];
                count++;
            }
            if(sum2==sum) {
                array=array.substring(1);
                System.out.println(array+"="+sum);
                int[] ans = new int[count];
                for(int k=0;k<count;k++) {
                    ans[k] = arr2[k];
                }
                return ans;
            }
        }
    }
    return null;
}
```